JURNAL INFORMATIKA Vol. 5, No. 2, Nopember 2004: 78 - 85

PEMBUATAN *PROTOTYPE* ANIMASI TIGA DIMENSI DENGAN MENGGUNAKAN *NETIMMERSE LIBRARY*

Rudy Adipranata
Fakultas Teknologi Industri, Jurusan Teknik Informatika, Universitas Kristen Petra
e-mail : rudya@petra.ac.id

ABSTRAK: Animasi tiga dimensi adalah salah satu bidang pada teknik komputer yang berkembang dengan cepat dewasa ini. Berbagai macam teknik dan algoritma telah dikembangkan pada bidang tersebut. Begitu pula aplikasi untuk bidang tersebut mencakup area yang luas, sebagai contoh adalah perangkat lunak game, bidang kedokteran, desain arsitektur, iklan dan lain-lain.

Tulisan ini menjelaskan tentang pembuatan prototype animasi tiga dimensi dengan menggunakan NetImmerse library. Di mana penjelasan ini mencakup mulai dari pembuatan karakter, labirin, penggabungannya dengan program serta implementasi pendeteksian benturan antara karakter dengan labirin.

NetImmerse adalah satu dari library untuk animasi tiga dimensi yang menyediakan seperangkat peralatan, plugins, dan juga run-time libraries. Di samping itu, NetImmerse juga menunjang beberapa platform. Pembuatan karakter dilakukan dengan menggunakan 3D Studio Max yang merupakan peralatan bantu untuk pembuatan karakter guna animasi tiga dimensi. NetImmerse mempunyai plugin untuk 3D Studio Max, sehingga dengan mudah hasil karakter dapat diekspor ke format NetImmerse.

Kata kunci: Animasi Tiga Dimensi, NetImmerse.

ABSTRACT: *3D Animation is one of the computer engineering field that growth fastly now. A lot of new techniques and algorithms have been developed in that field. Its application covered wide area, for example game software, medical analysis, architecture design, advertising.and many more.*

This paper describes how to implement prototype of 3D Animation using NetImmerse library. It includes a discussion on how to create character model and to implement it on NetImmerse library, to create a maze and to implement collision detection features between character model and maze.

NetImmerse is one of the greatest 3D engine library that provided a comprehensive set of tools, plugins, and run-time libraries. It provides end-to-end support to many plafform. Character model is created using 3D Studio Max Software, the one of most powerful 3D Animation Software. NetImmerse has plugin of 3D Studio Max Software, so can easily export the character have created to the NetImmerse format.

Keywords: *3D Animation, NetImmerse.*

1. PENDAHULUAN

Pada area animasi tiga dimensi, terdapat banyak teknik dan algoritma baru yang dikembangkan akhir-akhir ini. Dan banyak terdapat pula peralatan bantu yang mempunyai kemampuan untuk membuat animasi tiga dimensi. NetImmerse yang dibuat oleh perusahaan Numerical Design Limited (NDL) adalah salah satu paket yang terdiri dari peralatan bantu (*tools*), *plugins* dan *run time libraries* yang dapat digunakan untuk membuat aplikasi animasi tiga dimensi. NetImmerse merupakan paket bantu perangkat lunak dalam bahasa C++ yang berorientasi obyek dan mempunyai tujuan untuk menghemat waktu dan biaya bagi para pembuat *game* tiga dimensi dan aplikasi tiga dimensi untuk mengembangkan *game* ataupun

aplikasi tersebut. NetImmerse menyediakan segala kebutuhan, fasilitas dan efek yang dibutuhkan oleh pengembang aplikasi tiga dimensi. Sebagai tambahan, NetImmerse tidak hanya mendukung pengembangan aplikasi pada komputer, tetapi juga pada mesin-mesin *game* seperti Xbox, Playstation 2 serta GameCube. Untuk penggunaan pada komputer, sistem operasi yang didukung oleh NetImmerse adalah keluarga Windows yaitu Windows 98, Me, 2000 dan XP). Pembuatan aplikasi pada artikel ini menggunakan NetImmerse 4.2.

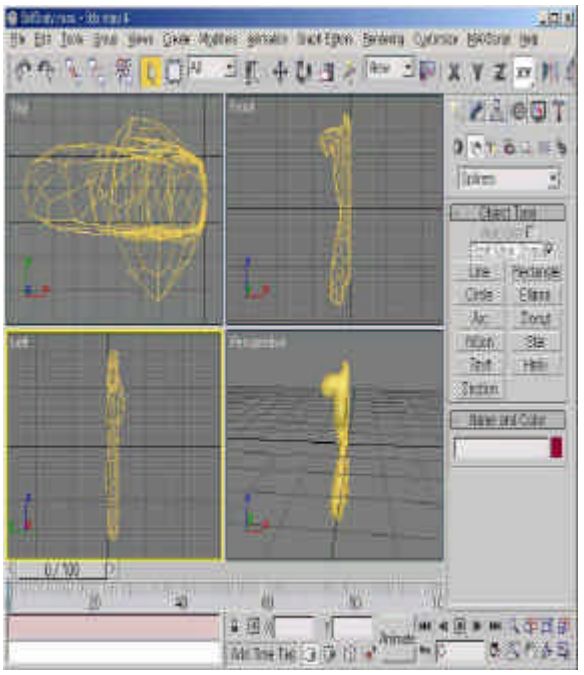
Artikel ini menjelaskan tentang pembuatan prototype animasi tiga dimensi dengan memanfaatkan NetImmerse. Adapun prototype animasi ini adalah sebuah karakter yang dapat digerakkan di dalam labirin (*maze*). Langkah pertama adalah penjelasan mengenai pembuatan model karakter tiga dimensi dan penggabungan karakter tersebut ke dalam aplikasi dengan menggunakan NetImmerse *library*. Langkah kedua adalah pembuatan model labirin dan penggabungan dengan karakter yang telah terbuat sehingga karakter dapat bergerak di dalam labirin. Cara pembuatan *collision detection* antara karakter dengan labirin juga akan diimplementasikan pada tahap ini.

Pembuatan model karakter dan labirin menggunakan perangkat lunak 3D Studio Max versi 4.2. Perangkat lunak tersebut adalah salah satu yang terbaik di bidang pembuatan karakter tiga dimensi. Dengan menggunakan 3D Studio Max, dapat dibuat sebuah model karakter dan menganimasikannya. NetImmerse mempunyai *plugins* untuk 3D Studio Max, sehingga tidak terjadi kesulitan untuk mengekspor format 3D Studio Max ke format NetImmerse.

2. PEMBUATAN MODEL KARAKTER

Penggunaan 3D Max Studio 4.2 untuk pembuatan model karakter mempunyai dua metode, yaitu low *polygon character modelling* dan *patch character modelling*. Aplikasi ini menggunakan *low polygon character modelling*. Metode ini dimulai dengan pembuatan obyek sederhana seperti garis yang kemudian dikombinasikan sehingga membentuk struktur tubuh dari karakter. Setelah selesai melakukan pembentukan struktur tubuh, dilakukan metode *extrude* untuk membuat bentuk yang semula terdiri dari garis-garis menjadi model tiga dimensi. Dan lebih lanjut, untuk membuat model tiga dimensi tersebut menjadi lebih halus, dapat dilakukan metode *mesh smooth*. Metode *mesh smooth* ini adalah salah satu fasilitas terbaik yang ada di 3D Max Studio untuk membuat poligon yang halus dengan berbasiskan poligon sederhana.

Model karakter yang dibuat pada aplikasi ini adalah karakter manusia, dimana hasil setelah pembuatan garis dan melakukan metode seperti dijelaskan di atas, dapat dilihat pada gambar 1 di bawah ini.

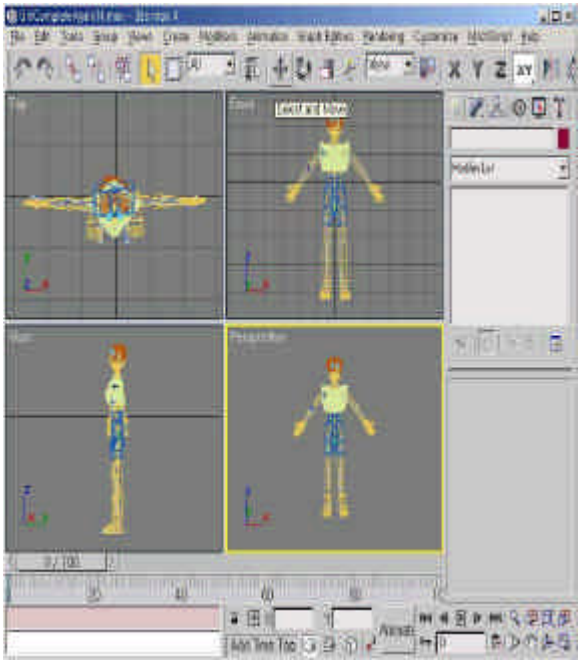


Gambar 1. Setengah Model Karakter

Untuk pertama kali, hanya diperlukan pembuatan setengah model karakter karena setelah selesai akan bisa dilakukan *mirroring* terhadap sumbu z untuk pembentukan karakter penuh.

Untuk kepala, tangan, dan kaki dari karakter dibuat secara terpisah dengan menggunakan metode yang sama dan

pada akhirnya nanti akan digabung dengan tubuh karakter. Walaupun dimungkinkan untuk membuat secara utuh sekaligus tapi dengan pembuatan secara terpisah ini akan mendapat keuntungan yaitu pengaturan yang mudah. Dan sama dengan pembuatan tubuh, kepala juga dibuat hanya setengah dan setelah selesai dilakukan mirroring untuk membuatnya menjadi satu kepala yang utuh. Hasil karakter secara lengkap dapat dilihat pada gambar 2 berikut ini.

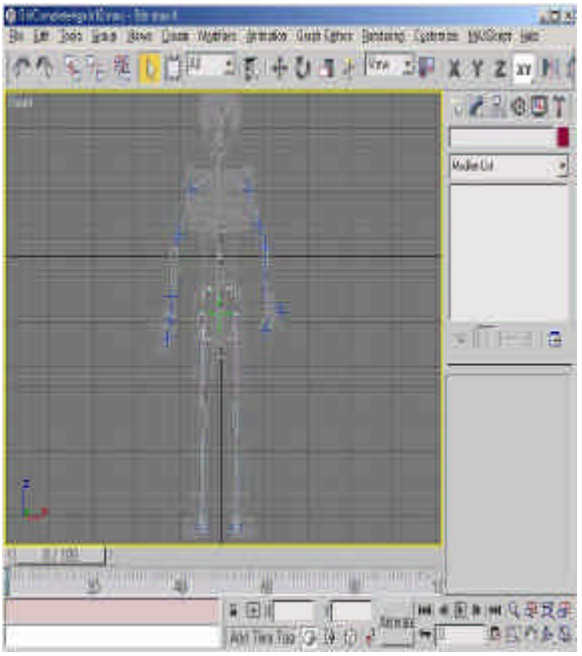


Gambar 2. Model Karakter Lengkap

3. PEMBUATAN ANIMASI PADA KARAKTER

Setelah menyelesaikan pembuatan karakter, langkah berikut adalah membuat animasi pada karakter tersebut. 3D Max Studio mempunyai fasilitas untuk dapat membuat animasi secara mudah. Terdapat dua metode untuk membuat animasi pada karakter, yaitu dengan menggunakan metode *bone* (tulang) atau menggunakan metode *biped*. Pada aplikasi ini, digunakan metode *bone* untuk membuat animasi.

Dengan menggunakan metode *bone*, langkah pertama ialah membuat *bone* pada karakter yang akan dianimasikan. *Bone* ini dibuat pada titik-titik yang serupa dengan tulang manusia, yaitu pada penghubung antara bagian-bagian yang dapat bergerak, harus dibuat *bone* yang terpisah. Setelah selesai membuat *bone*, maka harus dilakukan perintah Inverse Kinematics (IK) Solver. Tujuan IK Solver ini adalah untuk memberikan pengontrolan guna menganimasikan karakter. *Bone* yang telah dibuat pada karakter dapat dilihat pada gambar 3 di bawah ini.

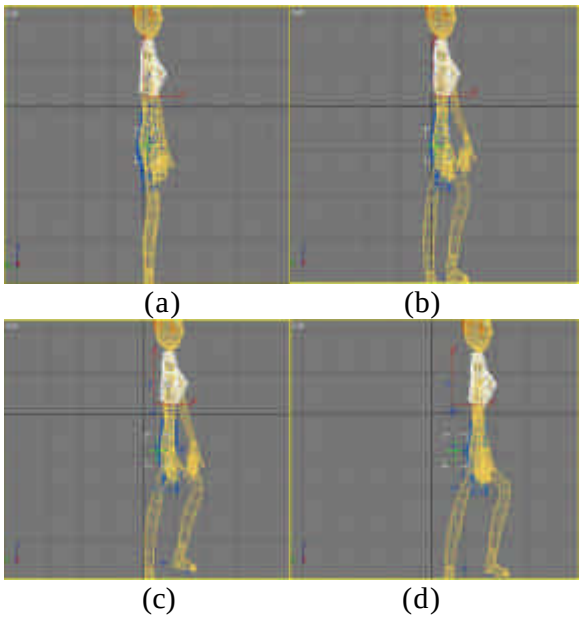


Gambar 3. Karakter Beserta Bone Yang Telah Dibuat

Konsep animasi pada komputer adalah penampakan sejumlah *frame* yang berisi gambar secara berurutan dengan kecepatan tertentu. Setiap *frame* berisi satu buah gambar yang tidak bergerak tetapi dikarenakan *frame* tersebut ditampilkan secara berurutan dengan kecepatan tertentu, mata manusia akan melihat gambar tersebut bergerak atau merupakan animasi.

Sehubungan dengan konsep tersebut, maka langkah selanjutnya untuk menganimasikan karakter adalah dengan mendefinisikan tiap *frame*. Pada 3D Max Studio, terdapat alat bantu untuk mendefinisikan setiap *frame* secara mudah dengan menggunakan time slider pada bagian bawah dari *view port* pada 3D Max Studio. Pertama kali adalah mengatur

waktu yang diinginkan untuk menampilkan *frame*, dan kemudian meletakkan posisi karakter pada waktu yang telah ditentukan tersebut. Langkah ini diulangi secara terus menerus, makin banyak *frame* yang didefinisikan, maka makin halus karakter tersebut akan bergerak atau melakukan animasi. Beberapa contoh dari *frame* karakter bergerak ditunjukkan pada gambar 4 dibawah ini.



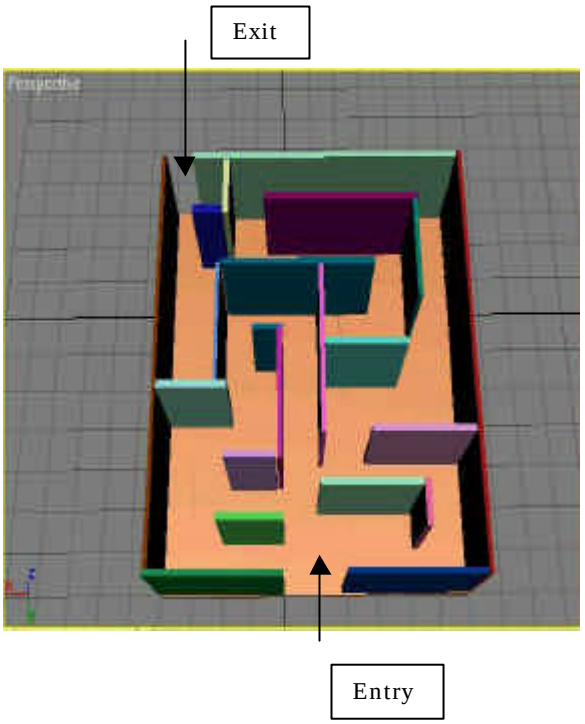
Gambar 4. Urutan Frame Pada Animasi Karakter

Setelah selesai melakukan pembuatan karakter dan animasinya, maka langkah terakhir yang dilakukan pada 3D Studio Max adalah mengekspor ke dalam format NIF file, yaitu format yang dikenali oleh NetImmerse, yang selanjutnya file ini akan digunakan dengan memanfaatkan NetImmerse *library*.

4. PEMBUATAN LABIRIN / *MAZE*

Pembuatan labirin ini juga dilakukan dengan menggunakan 3D Max Studio dengan langkah yang mirip dengan pembuatan model karakter. Langkah pertama untuk pembuatan labirin ini ialah dengan membuat obyek yang berbentuk persegi panjang yang nantinya obyek ini akan menjadi tembok dari labirin. Setelah membuat obyek dengan bentuk persegi panjang, dapat dilakukan metode *extrude* untuk membuat persegi panjang tersebut menjadi balok.

Labirin haruslah mempunyai pintu masuk dan pintu keluar. Lokasi pertama karakter nantinya adalah pada pintu masuk, dan nantinya karakter tersebut dapat digerakkan untuk menuju ke pintu keluar dari labirin. Gambar labirin yang digunakan pada aplikasi ini adalah sebagai berikut.



Gambar 5. Labirin

Setelah pembuatan labirin selesai, maka seperti pembuatan karakter, labirin ini juga harus dieksport ke format NIF file sehingga bisa dibaca oleh NetImmerse *library*.

5. HASIL PEMROGRAMAN ANIMASI KARAKTER DAN LABIRIN

Pemrograman aplikasi ini dilakukan dengan menggunakan bahasa pemrograman Microsoft Visual C++ 6.0. NetImmerse menyediakan *general class* yaitu NiApplication yang dapat digunakan sebagai *base class* dari aplikasi. NiApplication adalah *framework* alternatif yang dapat menggantikan *framework*

lain yang lebih kompleks seperti Microsoft Foundation Class (MFC).

NiApplication membungkus *entry point* pada pemrograman Win32 (WinMain) sehingga menjadi sebuah fungsi yang statis, yang diatur pada file NiAppMain.cpp. Berbagai macam *callbacks* telah disediakan untuk mengijinkan aplikasi mengendalikan pembuatan *window* yang termasuk juga di dalamnya adalah pembuatan menu dan *status bar*. Fungsi utama yang tersedia akan membuat sebuah window yang digunakan oleh aplikasi. *Messages* atau *events* ditangani oleh *class* NiEventHandler. *Class* NiEventHandler ini menyediakan fungsi di NiEventHandler.cpp untuk membaca dan meneruskan *message* dari sistem operasi. Sejumlah *virtual message handlers* tersedia pada NiAppEvents.cpp yang akan menangani hampir semua *messages* dan dapat di *override* oleh aplikasi dengan menggunakan *class* NiApplication.

NiApplication mempunyai tiga *members* yang diturunkan dari NiObject dan tiga *virtual functions* yang digunakan untuk membuat obyek pada NetImmerse yaitu : *scene*, kamera dan *renderer*. Dalam aplikasi ini semua variabel dan metode akan di *override*. *Member* dari sebuah scene adalah m_spScene, dimana m_p Scene ini merupakan bagian teratas dari *scene graph*. Untuk mengalokasikan *node* ke m_spScene dapat digunakan *virtual function* CreateScene. *Member* dari kamera adalah m_spCamera, dimana m_spCamera ini berfungsi sebagai kamera utama untuk melihat *scene graph*. Untuk dapat menggunakan kamera ini maka harus terlebih dahulu dibuat obyek NiCamera dengan karakteristik sesuai dengan yang diinginkan, yaitu dengan menggunakan *virtual function* CreateCamera. Sedangkan member untuk *renderer* adalah m_spRenderer yang berfungsi sebagai *renderer* utama yang digunakan untuk me-*render* tampilan pada layar monitor. Untuk menggunakan *renderer*, diperlukan obyek NiDisplay dan NiRenderer yang dibuat dengan menggunakan *virtual function* CreateRenderer.

Tiga buah *virtual function* yang telah dijelaskan di atas yaitu CreateScene, CreateCamera dan CreateRenderer dipanggil di dalam fungsi Initialize. Fungsi Initialize ini adalah fungsi pertama yang dipanggil jika aplikasi dijalankan. Untuk implementasi pada program, dapat dilihat pada kode di bawah ini :

```
bool Rudy3DApp::Initialize ()
{
   if ( !CreateRenderer() )
      return false;
   if ( !CreateScene() )
      return false;
   CreateCamera();
   CreateLight();
   m_spCamera->
SetRenderer(m_spRenderer);
   m_spCamera->
SetScene(m_spScene);
   m_spScene->Update(0.0f);
   m_spScene->UpdateProperties();
   m_spScene->UpdateEffects();
      return true;
}
```

Program ini melakukan *override* terhadap fungsi asal CreateScene pada NiApplication. Pada CreateScene, terdapat fasilitas untuk membuat m_sp Scene yang merupakan bagian teratas dari semua yang berhubungan dengan *scene graph*. Aplikasi ini juga membuat obyek m_spControlNode yang bertipe *pointer* dari NiNode. m_spControlNode ini digunakan sebagai *base class* dari semua pengendalian karakter dan kamera. Untuk mengendalikan karakter, dibuat sebuah obyek m_spBoy yang juga bertipe *pointer* dari NiNode. Cuplikan kode untuk mengambil (*loading*) obyek karakter dari file NIF dan ditempatkan dalam aplikasi adalah sebagai berikut :

```
// Get character data from file and put into
memory (stream)
   if (! stream.Load(
ConvertMediaFilename("MonsterFinal.nif")))
            return false;
   assert(stream.GetObjectCount() == 1);
Get character data from memory (stream) and
put into pObject
   NiObject* pObject =
         stream.GetObjectAt(0);
```

```
   NiNodePtr spModel = 0;
Change data type of NiObject to NiNode
   spModel = NiDynamicCast(NiNode,
                                pObject);
   assert(spModel);
// Assign m_spBoy with character data in
spModel
   m_spBoy = (NiNode*)spModel->
             GetObjectByName("center");
// Attach m_spBoy to base class
gm_spControlNode
   m_spControlNode->
             AttachChild(m_spBoy);
```

Setelah melakukan pembuatan dan pengambilan (*loading*) data karakter, langkah berikutnya adalah pengambilan data file labirin. Aplikasi ini mendefinisikan satu obyek untuk mengendalikan file labirin yaitu spRooms. Tipe obyek ini juga *pointer* dari NiNode dimana setelah melakukan pembuatan obyek tersebut, harus melakukan pengambilan data dari file NIF. Kode untuk melakukan pengambilan data labirin adalah sebagai berikut :

```
// Get maze data from file and put into
memory (stream)
   if (! stream.Load(
NiApplication::ConvertMediaFilename("Rudy
Maze4.nif")))
   {
      MessageBox(NULL, "Error loading file.",
"Error", MB_OK);
      return 0;
   }
  assert( NiIsKindOf(NiNode,stream.
GetObjectAt(0)) );
NiNodePtr spModel = (NiNode*)
stream.GetObjectAt(0);
```

Langkah terakhir untuk mengakhiri pengambilan data karakter dan labirin adalah pembuatan kamera. Obyek kamera ini dikendalikan oleh satu obyek yaitu m_spCamera yang bertipe *pointer* dari NiCamera. Setelah melakukan pembuatan kamera, maka posisi kamera harus ditempatkan di belakang karakter dan kemudian dimasukkan ke m_sp ControlNode. Kamera diletakkan dibelakang karakter agar pengguna yang menjalankan karakter mempunyai pandangan yang sama dengan pandangan karakter yaitu ke arah depan. Kode untuk pembuatan kamera ini adalah sebagai berikut :

```
   m_spCamera = new NiCamera;
   assert(m_spCamera);
   NiBound sph = m_spScene->
             GetWorldBound();
   NiCamera::Frustum fr = m_spCamera->
             GetViewFrustum();
   fr.m_fNear = 1.0f;
   fr.m_fFar =  2.0f * sph.GetRadius();
   m_spCamera->SetViewFrustum(fr);
   // Attach scene to camera
   m_spCamera->
SetRenderer(m_spRenderer);
   m_spCamera->SetScene(m_spScene);
  sph = m_spBoy->GetWorldBound();
  // Orient camera slightly above and behind
the controlled character
   NiMatrix3  rotX, rotY, rot;
   rotX.MakeXRotation( -NI_PI * 0.5f );
   rotY.MakeYRotation( -NI_PI / 10.0f );
   rot = rotY * rotX;
   m_spCamera->SetRotate(rot);
   m_spCamera->
  SetTranslate(-2.2f * sph.GetRadius(), 0,
      2.2f * sph.GetRadius());
   m_spControlNode->
       AttachChild(m_spCamera);
```

Setelah selesai melakukan pengambilan data karakter dan labirin, langkah berikutnya adalah membuat karakter tersebut bergerak jika menerima *input* berupa penekanan tombol *keyboard*. Gerakan karakter meliputi gerakan lurus ke depan, belakang, memutar kiri dan kanan. Metode untuk melakukan gerakan ke depan dan belakang adalah dengan melakukan translasi terhadap sumbu X positif (ke depan) dan sumbu X negatif (ke belakang). Untuk memutar ke kiri dan kanan, metode yang dipakai adalah dengan melakukan rotasi terhadap sumbu Y negatif dan positif.

Translasi dan rotasi ini hanya perlu dilakukan pada obyek m_spControlNode karena kamera dan karakter semuanya terhubung ke m_spControlNode, yang berarti setiap perubahan yang terjadi pada m_spControlNode akan berlaku pada anaknya juga. Dan juga karena *view* kamera adalah m_spScene, maka otomatis tampilan scene juga akan berubah jika terjadi gerakan. Jadi transformasi dan rotasi terjadi pada karakter dan kamera

sekaligus. Kode untuk animasi karakter ditempatkan pada fungsi OnIdle. Program ini menggunakan NiTimeController untuk mengendalikan animasi karakter. NiTimeController adalah *base class* untuk semua pengendalian animasi dimana *class* tersebut menyediakan fungsi dasar untuk pewaktuan. Prosedur untuk menggunakan NiTimeController adalah sebagai berikut :

```
for (pControl = pObject->
      GetControllers(); pControl;
      pControl = pControl->GetNext())
    {
      pControl->
SetCycleType(NiTimeController::LOOP);
    }
```

Gambar 6 adalah struktur dari semua *pointer* obyek pada *scene graph* .

6. PEMROGRAMAN DETEKSI BENTURAN

Pendeteksian benturan antara obyek karakter yang satu dengan karakter yang lain atau dengan labirin dapat dilakukan dengan mengidentifikasi perpotongan antara obyek geometri yang satu dengan obyek geometri yang lain. Dan jika terdeteksi adanya perpotongan, maka harus dihasilkan respon yang sesuai. Untuk mengimplementasikannya, langkah pertama adalah melakukan pembuatan *bounding sphere* untuk setiap obyek yang berada pada *scene*. NetImmerse memupunyai *class* yang dapat digunakan untuk membuat sebuah *bounding volume*, salah satunya adalah NiCapsuleBV.

Jika terdapat gerakan yang lebih kompleks, maka diperlukan tipe bounding volume yang lebih kompleks pula sehingga dapat mendeteksi benturan dengan lebih tepat. Kode untuk pembuatan obyek dari class NiCapsuleBV dan prosedur untuk pengecekan benturan adalah sebagai berikut :

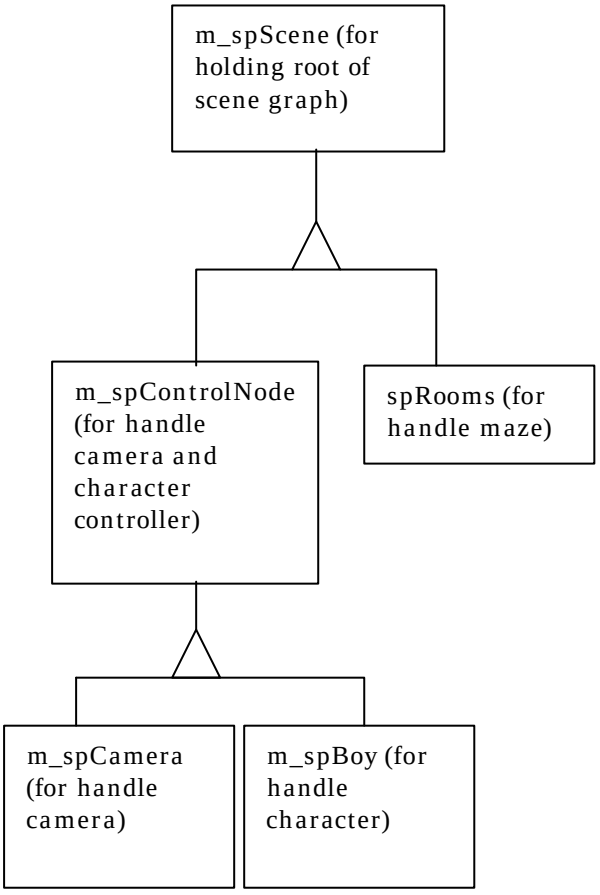
```
// creating bounding volume
  NiCapsuleBV* pBV = new NiCapsuleBV
(0.0f, 20.0f, NiPoint3(0, 0, 0),
        NiPoint3::UNIT_Z);
  assert(pBV);
  pNode->SetModelSpaceABV(pBV);
```

```
  NiNode*    pCollider;
  NiAVObject* pCollidee;
  NiPoint3*   colliderNormal;
  NiPoint3*   collideeNormal;
  const char* pName = intersect.pkRoot0->
      GetName();
  if (pName && (! strcmp(pName,
      "center")))
  {
    pCollider = (NiNode*)
intersect.pkRoot0;
     pCollidee = intersect.pkRoot1;
    colliderNormal = &intersect.kNormal0;
    collideeNormal = &intersect.kNormal1;
  }
  else
  {
   pCollider = (NiNode*)intersect.pkRoot1;
     pCollidee = intersect.pkRoot0;
    colliderNormal = &intersect.kNormal1;
    collideeNormal = &intersect.kNormal0;
  }

  Interact* pThis = (Interact*) pCollider->
      GetCollideCallbackData();
  ColliderRecord& record = pThis->
        m_ColliderRec;
  record.bCollided = true;
  // Keep track of the minimum time at
which the collider will collide
  // with another object. We want the update
routine to handle the
  // first object the collider will hit.
 if (intersect.fTime < record.fIntersectTime)
  {
    record.fIntersectTime = intersect.fTime;
    record.intersectPoint = intersect.kPoint;
 record.colliderNormal = *colliderNormal;
 record.collideeNormal = *collideeNormal;
  }
   // Store the collidee information for ALL
collisions,
  // order of storage is not important as there
needs to be no
  // later correlation between normals &
collidees.
 CollideeRecord* pCR = pThis->
 m_CollideeRecs.GetAt
      (record.uiNumCollisions++);
  assert(pCR);
  pCR->bCollided = true;
  pCR->normal = *collideeNormal;
  return
NiCollisionGroup::CONTINUE_COLLISIONS
;
```

Kode di atas ini akan selalu dipanggil jika terjadi perubahan pada obyek seperti

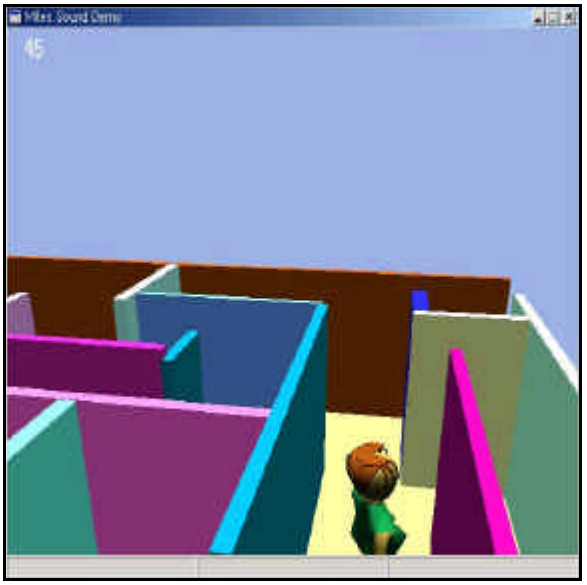
gerakan obyek dan melakukan pengecekan terhadap obyek apakah terdapat benturan dengan obyek yang lain. Jika terdapat benturan, maka obyek tidak dapat bergerak sedangkan jika tidak terdapat benturan, maka gerakan obyek dapat dilanjutkan.



Gambar 6. Stuktrur Obyek Pada Scene Graph

7. HASIL DAN KESIMPULAN

Hasil dari penelitian ini adalah sebuah prototype animasi tiga dimensi dimana animasi ini terdiri dari sebuah karakter yang ditempatkan di dalam sebuah labirin. Karakter ini dapat digerakkan ke depan, belakang, memutar ke kanan dan kiri dengan melakukan penekanan tombol keyboard. Jika pada saat bergerak ke depan atau belakang, karakter membentur dinding labirin, maka karakter tersebut akan tetap, tidak melakukan gerakan. Tampilan dari prototype ini dapat dilihat pada gambar 7 di bawah ini.



Gambar 7. Tampilan Prototype Animasi Tiga Dimensi

Sebagai kesimpulan, dengan menggunakan NetImmerse *library* akan mempermudah pembuatan aplikasi animasi tiga dimensi baik untuk melakukan pembacaan format file, sinkronisasi antar obyek dan penampilan (*rendering*) pada layar monitor. Aplikasi ini dapat digunakan untuk berbagai macam keperluan seperti pembuatan *game*, *virtual reality*, dan lain-lain.

DAFAR PUSTAKA

1. Alan Watt, *"Fundamentals of Three-Dimensional Computer Graphics"*, Addison-Wesley Publishing Company – 1989.

2. Autodesk, Inc, *"3D Studio Max 4.1 Tutorials"*.

3. Numerical Design, Ltd, *"NetImmerse 4.2 Evaluation Help"*.